University of Saskatchewan
Department of Computer Science

Cmpt 330
Final Examination

December 13, 2000

Time: 3 hours
Total Marks: 100

Professor: A. J. Kusalik
Closed Book†

Name: _____

Student Number: _____

Directions:

Answer each of the following questions in the space provided in this exam booklet. If you must continue an answer (e.g. in the extra space on the last page, or on the back side of a page), make sure you clearly indicate that you have done so and where to find the continuation.

Ensure that all answers are written legibly; no marks will be given for answers which cannot be deciphered. Where a discourse or discussion is called for, please be concise and precise. Do not give "extra answers". Extra answers which are incorrect may result in your being docked marks.

Use of calculators during the exam is not allowed.

If any question requires an assumption as to a particular operating system context, assume UNIX as manifest by NetBSD. If you find it necessary to make any other assumptions to answer a question, state the assumption with your answer.

Marks for each major question are given at the beginning of that question. The last page of the exam contains supplemental information which may be of use in answering some of the questions.

Good luck.

For marking use only:

A.	____/11	E.	____/6	I.	____/10
B.	____/11	F.	____/5	J.	____/23
C.	____/8	G.	____/8		
D.	____/12	H.	____/6		

Total: ____/100

† Closed book, except for one optional 8.5×11 inch quick reference sheet ("cheat sheet") of the student's own compilation.

A. *(11 marks)*

For each of the statements below indicate whether it is **true** ("T") or **false** ("F").

F One of the problems with file transfers on UUCP was that routing had to be (explicitly) specified.

F It is not possible for the major unit number and minor unit number (for a special file) to be the same value; e.g. one could never have a special file for a device (on UNIX) where both the major unit and minor unit numbers were 12.

F To improve robustness in the BSD FFS, the number of cylinders per cylinder group is varied as one "moves in" towards the center of the disk drive.

F After a `fork()` call, the parent and child cannot communicate. I.e. after a `fork()`, there is no way, for example, for the child to get information to the parent.

T `time(3)` is a kernel function that can be used to obtain the current time-of-day from which measurements such as elapsed time for a program can be determined.

T The three types of processing provided for terminal streams by UNIX are "cooked", "cbreak", and "raw".

F Disk partitioning is supported by a combination of hardware and software. The partition table is passed by the device driver to the disk controller, which subsequently does translation of "partition plus sector" addresses.

T The "magic number" for a file (used by the `file(1)` program for identifying the contents of a file) can occur at any location in the file, though it is often near the beginning.

T To be able to write information to the middle of a file (e.g. to `lseek()` to an address in the middle of a file and then start `write()`-ing), the file must have a "hole" at the location being written to. Otherwise, if there is already information stored at that location, the `write()` cannot be allowed because the existing information would be over-written.

T An electronic mail message can be considered to be a type of datagram; i.e. a unit of communication sent via a connectionless protocol.

T The domain name system (DNS) provides a distributed database service that supports dynamic retrieval of information about the Internet name space (such as mappings to and from IP addresses).

B. *(2+4+5=11 marks)*

Each of the following questions require very short, precise answers.

1. The file `/usr/include/ufs/ufs/dinode.h` contains, in part

```
typedef int32_t ufs_daddr_t;
#define NDADDR 12               /* Direct addresses in inode. */
#define NIADDR 3                /* Indirect addresses in inode. */

struct dinode {
    u_int16_t   di_mode;        /*   0: IFMT, permissions */
    int16_t     di_nlink;       /*   2: File link count. */
```

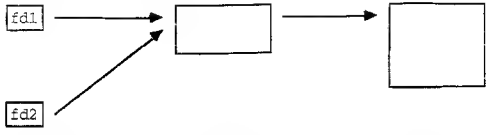
```
    union {
        u_int16_t oldids[2];    /*   4: Ffs: user and group ids. */
        ino_t     inumber;      /*   4: Lfs: inode number. */
    } di_u;
    u_int64_t   di_size;        /*   8: File byte count. */
    int32_t     di_atime;       /*  16: Last access time. */
    int32_t     di_atimensec;   /*  20: Last access time. */
    int32_t     di_mtime;       /*  24: Last modified time. */
    int32_t     di_mtimensec;   /*  28: Last modified time. */
    int32_t     di_ctime;       /*  32: Last inode change time. */
    int32_t     di_ctimensec;   /*  36: Last inode change time. */
    ufs_daddr_t di_db[NDADDR];  /*  40: Direct disk blocks. */
    ufs_daddr_t di_ib[NIADDR];  /*  88: Indirect disk blocks. */
    u_int32_t   di_flags;       /* 100: Status flags (chflags). */
    int32_t     di_blocks;      /* 104: Blocks actually held. */
    int32_t     di_gen;         /* 108: Generation number. */
    u_int32_t   di_uid;         /* 112: File owner. */
    u_int32_t   di_gid;         /* 116: File group. */
    int32_t     di_spare[2];    /* 120: Reserved; cur. unused */
};

/*
 * The di_db fields may be overlaid with other information for
 * file types that do not have associated disk storage. Block
 * and character devices overlay the first data block with their
 * dev_t value. Short symbolic links place their path in the
 * di_db area.
 */
#define di_rdev         di_db[0]
#define di_shortlink    di_db
#define MAXSYMLINKLEN   ((NDADDR + NIADDR) * sizeof(ufs_daddr_t))
```

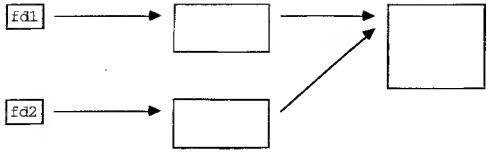
Therefore, what is the maximum length of a symbolic link for a (FFS) file system on this machine?

2. Consider the following three diagrams:

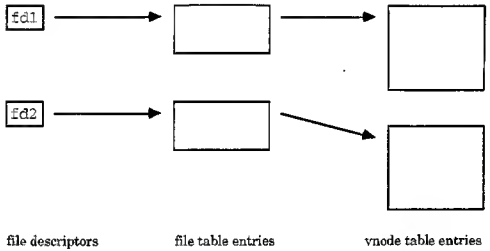
(a)



(b)



(c)



Indicate which diagram above corresponds to the situation after each of the following sequences of operations.

i. b
```
fd1 = open( pathname, ... );
fd2 = open( pathname, ... ); /* open the same file */
```

ii. a
```
fd1 = open( pathname, ... );
fd2 = dup( fd1 );
```

iii. c
```
fd1 = open( pathname, ... );
dup2( fd1, fd2 );
```

iv. ___

```
fd = open( pathname, ... );
if( fork() == 0 ) {
... /* fd2 in the diagram is the child's copy of fd above */
} else {
... /* fd1 in the diagram is the parent's copy of fd above */
}
```

3. A client-server architecture is commonly used for implementing and accessing network services. Two different processes (communicants), one called the server and one called the client, are involved. The exchange between the server and the client can be connectionless or connection-based. Assuming the latter, the server will typically perform the following operations:

   a) create a (socket) endpoint

   b) assign an address (to the socket)

   c) tell the kernel that it (the server) is ready to accept connections

   d) wait for a connection

   e) transfer data

   f) terminate the connection

   while the client performs

   f) create a (socket) endpoint

   g) establish a connection

   h) transfer data

   i) terminate the connection

   Consider each of the following five UNIX system calls. For each, state which of the operations above it is used for. To specify your answer you can use just the letter labelling an operation, if you wish. If more than one operation above can be performed by one of the system calls, you need only specify one.

   ___ *bind()*

   ___ *accept()*

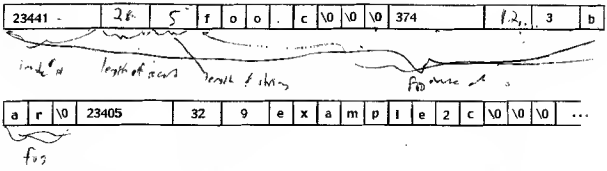
   ___ *send()*

   ___ *connect()*

   ___ *listen()*

C. (8 marks)

The following diagram represents a portion of a directory file under the BSD "Fast File System". Complete the diagram as if the file `bar` had existed, and was just deleted. Indicate the extent of the directory entry for file `foo.c` and label all fields within it. Also, fill in each blank field within the diagram with the correct value. You can assume that the directory entry for `foo.c` begins the chunk (directory block) and that the directory entry for `example2c` is valid (i.e. that file `example2c` exists in the directory in question).



Assume for this question that directory entries are defined by the following `struct` definition:

```
struct  direct {
    u_int32_t d_ino;                    /* inode number of entry */
    u_int16_t d_reclen;                 /* length of this record */
    u_int16_t d_namlen;                 /* length of string in d_name */
    char      d_name[MAXNAMLEN + 1];/* name with length <= MAXNAMLEN */
};
```

Note that this `struct` definition is slightly different from that in NetBSD 1.5.

D. (12 marks)

For each of the following pairs of underlined terms, indicate whether or not they are synonymous (mean the same thing). If they mean different things, contrast the two terms and explain how their meanings differ. If the two terms mean the same thing, give a definition; i.e. explain their single meaning. You may use examples to illustrate your point(s).

1. child process and client process

2. socket and pipe

3. fragment block and fragment

4. free-space reserve and free space map

E. (6 marks)

A Cmpt330 student is working on an assignment for her Cmpt330 class. The program the student is to write should disable the generation of SIGINT signals by the "interrupt" character, and set the EOF character to ^B (control-B). After several hours work, the student has achieved the following program:

```
#include <termios.h>
#include <unistd.h>

int main(void)
{
   struct termios  *term;
   long  vdisable;
   if (isatty(STDIN_FILENO) == 0) { /* true or false */
      perror("isatty");
      exit( 1 );
   }

   if ( (vdisable = fpathconf(STDIN_FILENO, _PC_VDISABLE)) < 0) {
      perror("fpathconf or _POSIX_VDISABLE");
      exit( 2 );
   }

   if (tcgetattr(STDIN_FILENO, term) < 0) /* fetch tty state */ {
      perror("tcgetattr");
      exit( 3 );
   }

   term->c_cc[VINTR] = vdisable;        /* disable INTR character */
   term->c_cc[VEOF]  = 2;               /* make EOF be Control-B */
```